

# A Parameter Reduction Based Technique For Automatic Analysis of Database Systems

Ning Jiang, Kien A. Hua  
{njiang, kienhua}@cs.ucf.edu  
School of Electrical Engineering and Computer Science  
University of Central Florida  
Orlando, FL 32816

Morgan C. Wang  
cwang@mail.ucf.edu  
Department of Statistics  
University of Central Florida  
Orlando, FL 32816

## Abstract

*Analyzing database system performance is a difficult task. Traditional tools rely mainly on graphing to analyze data in an ad-hoc manner. Determining the right parameters to examine is often a lengthy process of educated guesswork; where many problems can run undetected. To cope with the enormous number of parameters (in the magnitude of thousands) that can affect system performance, we introduce a two-phase parameter reduction technique. In Phase 1, variable clustering is applied to group correlated parameters together; and only one representative parameter is chosen from each cluster. In addition, parameters not highly correlated with any other parameter are also selected for further analysis. In Phase 2, regression tree analysis is applied to reduce the number of parameters further. The result is a set of parameters most sensitive to changes in the system resources. These parameters are used to construct a statistical model that describes the causal behavior of the system. This model and the correlations obtained from the parameter selection process provide the user with insight to fine tune the system performance. To assess the effectiveness of this approach, we implement an automatic analysis tool, and use it to analyze different configurations of a TPC-C (Transaction Processing Performance Council benchmark) database system. The experimental results indicate the effectiveness of the proposed technique.*

## 1. Introduction

Solving database performance problems [ORCLT99] is becoming more and more crucial with the

explosive growth in the scale and complexity of database systems. Many E-commerce websites are using database technology to provide online transaction processing (OLTP) and other services. Good performance of the database system is vital to their existence.

Generally speaking, the performance of a database system can be tuned [ORCLT99] [T2000] at two levels. High level tuning involves improving logical designs to achieve better overall performance. Examples include modifying the design of schema, writing more efficient applications, and adopting better data access paths. Low level tuning mainly deals with the internal parameters of the database and operating systems and the system hardware (processors, memory, disks, network, etc.). At both levels, correct identification of the most critical problem of the system is key to the success of the tuning effort. Traditional approaches ([B01], [G01], [ORCLT99]) rely primarily on graphing and examining a small number of system parameters. Determining which system resources to investigate is at best a lengthy process of educated guesswork, where many problems can run undetected. To overcome these drawbacks, we propose a systematic approach that employs data mining ([AGIS92], [AS94], [AS96], [HS95]) techniques to thoroughly investigate all system parameters. To cope with the enormous number of parameters (usually in the magnitude of thousands), we introduce a 2-phase parameter reduction technique. In each phase, this scheme evaluates the correlation and interaction among various parameters, and retains only those, whose effect on the system behavior can be inferred from other parameters, for further consideration. The result of this process is a small list of system

parameters that have the most direct influence on system performance, and are complete in capturing every aspect of system behavior. These selected parameters are used to construct a statistical model to quantitatively measure the impact of each selected parameter on system performance. This model reveals the most critical parameters for tuning the system. Our experimental studies indicate that this tool proves to be very useful in analyzing various configurations of (but not limited to) an Oracle database system.

## 2. Problem Formulation

We first define the notations to facilitate our discussion. Let  $\alpha = \{a_1, a_2, \dots, a_n\}$  be the set of system parameters. A data collection tool typically samples these parameters at fixed time intervals. At each time interval,  $a_i$  has a value  $v_i$ . Thus, the output of a data collection tool is a performance data set organized as a time series  $\tau = \{x_1, x_2, \dots, x_s\}$ , where  $x_t = \langle t, v_{1t}, v_{2t}, \dots, v_{nt} \rangle$  denotes a vector of system parameter values measured at time  $t$ . In addition to the system parameters, we define  $\pi = \{c_1, c_2, \dots, c_m\}$  as a set of *system state descriptors* (e.g. number of processors, storage capacity, size of swap space in memory, etc.) These descriptors convey the current state or configuration of the system. When system resources are upgraded or modified, these descriptors change accordingly to reflect the changes in the system state. The problem of system tuning, therefore, is to revise these system descriptors (e.g., adding one more processor) in a way to achieve a better system state. A system state is considered better if we can realize a certain desired value for some of the system parameters (e.g., *%idle* is reasonably low). Since system tuning is typically needed after upgrades or changes in the system configuration, we often need to compare the values of the system parameters before and after the change in the system state in order to make tuning decisions. To support such discussion, we introduce  $U$  as a set of time series measured under different system states.

## 3. Parameter Selection

Parameter selection is the process of selecting a subset of original parameters<sup>1</sup> by eliminating redundant and uninformative ones. We observed that in system performance data, parameters are often correlated and interacting with each other. Such correlation and interaction render a good opportunity

---

<sup>1</sup> The parameter selection process only considers relationships among system performance parameters. System state descriptors are used in statistical model construction.

for parameter reduction. On the other hand, with the large number of parameters under investigation, it's more desirable to divide parameter selection into multiple phases, from coarse to fine. In the first phase, only first order correlations are taken into consideration. More complex relationships are investigated in the second phase.

### 3.1 Phase 1 - Parameter Selection Using Variable Clustering

First order correlation is considered in this phase. Clusters of highly correlated parameters are formed. One can then select a representative parameter from each cluster and remove others without losing too much information. In other words, the pruned parameters are represented by the specific selected parameter. One commonly used method to group highly correlated parameters is variable clustering procedures such as the VARCLUS procedure in the SAS system (1994).

The output of the VARCLUS procedure is a set of clusters. Parameters within the same cluster have higher correlation with each other whereas parameters in different cluster have relatively lower correlation. This information can be precious to system performance analysis, as explained in the motivation example in Section 2. Parameters are selected from each cluster according to the following steps:

1. In practice, some parameters of a cluster have relatively lower  $r^2$  values than others. Interactions are usually indicated. Thus,  $f$  should be selected for further analysis.
2. Otherwise, one parameter is selected to represent the cluster.

The set  $P$  of selected parameters has an important property.

**Property 1:** Consider any two parameters  $f_1, f_2 \in P$ , where  $P$  is the set of parameters generated by selecting parameters from the output clusters of VARCLUS procedure according to step 1 and 2. There is no first order correlation between  $f_1$  and  $f_2$ .

### 3.2 Phase 2 - Parameter Selection Using Regression Tree

In phase 2 of the technique, interaction between parameters is examined to further prune redundant parameters. Since most of parameters have numerical values, regression tree is employed to analyze the data. Parameter selection is based on the output set of regression trees.

### 3.2.1 Regression Tree Analysis

The algorithm *Regression\_Tree\_Analysis* is designed to capture the interaction between parameters. A regression tree [BFOS98] is grown for each parameter in  $P$ . Each regression tree contains all the parameters that are (statistically) important to predict a specific parameter. The set of parameters that are of statistical importance to a parameter  $u$  is denoted as  $SPLIT(u)$ .

Another important output of the algorithm is a predictability graph  $G=(V, E)$ .  $G$  is a directed graph. An edge  $e=<u, v>(e \in E \wedge u, v \in V)$  records the interaction information: if  $v \in SPLIT(u)$ , an edge is added from  $v$  to  $u$ . Definition and property of the predictability graph will be given in the next subsection.

### 3.2.2 K-Interactive Property

We first define the *predictability relation* between a pair of parameters. Next, the *predictability graph* is introduced to model interactions between various parameters. Finally, degree of interaction between parameters is measured in terms of the length of the path within the *predictability graph*.

**Definition 1:** Consider a set of parameters  $A$ . A *predictability relation*  $R$  is defined on  $A \times A$ .  $R = \{<f_1, f_2> \mid f_1, f_2 \in A \wedge f_2 \in SPLIT(f_1)\}$ . In particular,  $<f, f> \in R$ .

Obviously,  $R$  is reflexive and transitive. However  $R$  is not symmetric, i.e. if a parameter  $f_1$  is used in the construction of the tree with  $f_2$  set to be the target, it is not necessary that  $f_2$  be used to split internal nodes of the tree with  $f_1$  as target. We define a directed graph based on the definition of  $R$ .

**Definition 2:** A graph  $G=(V, E)$  ( $V=P$ ,  $E = \{<u, v> \mid vRu\}$ ) is a *predictability graph*.

Figure 1 gives an example of a *predictability graph*. In the graph, each vertex represents a parameter. Edges model the *predictability relation*. Note that self-loops are not included in the graph, although they are counted in the parameter selection algorithm.

**Definition 3:** A parameter  $f_1$  is *k-interactive* with another parameter  $f_2$  iff the length of the shortest path from  $f_1$  to  $f_2$  is  $k$ .

The *k-interactive* definition captures the degree of influence of one parameter to the other. Two parameters  $f_1$  and  $f_2$  are 0-interactive to each other iff  $<f_1, f_2> \notin E \wedge <f_2, f_1> \notin E$ . A parameter  $f_1$  is 1-interactive with another parameters  $f_2$  implies that  $f_1$  directly affects  $f_2$ . A parameter  $f_1$  is *k-interactive*

( $k \geq 2$ ) with another parameters  $f_2$  means  $f_1$  has no direct influence on  $f_2$ .

### 3.2.3 Parameter Selection Based On Tree Analysis Result

The *predictability graph* is utilized to further reduce parameters. This is achieved by generating a set of parameters that “covers” all the other parameters based on the *predictability relation*.

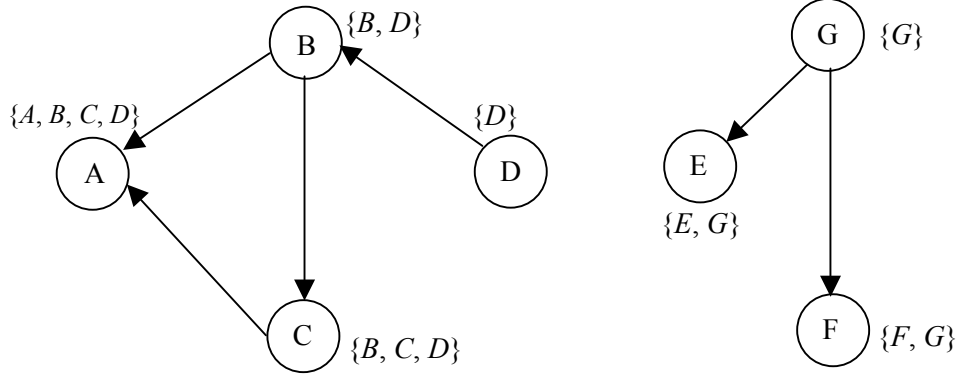
Consider a *predictability graph*  $G=(V, E)$ . In addition, we define a function  $Path(u, v)$ , which returns true when there’s a path from vertex  $u$  to vertex  $v$  and false otherwise. Basically, we want to generate a set of vertices (parameters)  $F$ , such that

- (1)  $F \subseteq V$ .
- (2)  $\forall u \in V(\exists v(v \in F \wedge Path(u, v)))$ .
- (3)  $\forall f_1, f_2 \in F$ ,  $f_1$  and  $f_2$  are 0-interactive to each other.
- (4)  $\neg \exists G(G \subset F \text{ and } G \text{ satisfies (2) and (3)})$ .

Transitivity of the *predictability relation* is the key to solve the problem. In essence, the problem is to first generate a *transitive closure* [S98] and then find the set of vertices that satisfy condition (1), (2), (3) and (4). In literature, there are already numerous algorithms [EK77, IRW93, NS93] that attack this problem. The idea lies in calculating for each node in  $V$  its *ancestor set* in which stores all the nodes that can “reach” it. Since the purpose of this research is not to find an optimal *ancestor set* generation algorithm, the naïve approach is adopted. The algorithm starts from each node  $v_i$  and performs a depth-first traverse. When a node is reached,  $v_i$  is propagated to its *ancestor set*. As an example, in Figure 1, each vertex’s *ancestor set* is marked beside it.

The next step is to generate the set  $F$  that satisfies requirement (1), (2), (3) and (4). In implementation, requirement (4) is relaxed. That is, the set  $F$  is not necessarily minimal. A greedy algorithm *Transitive\_Reduction* is designed. First, the algorithm sorts all vertices according to the size of their *ancestor set*. Second, nodes are selected iteratively in descending order of the size of their *ancestor sets*. Every time a node is selected, all the vertices in its *ancestor set* are marked as “covered”. This process is repeated until all the nodes are covered. Again, we take the graph in Figure 1 as an example. The selected vertex set is  $\{A, E, F\}$ .

The set  $F$  of selected parameters has an important property.



**Figure 1.** An Example of Predictability Graph

**Property 2:** Consider any two parameters  $f_1, f_2 \in F$ , where  $F$  is the output set of parameters of Algorithm *Transitive\_Reduction*.  $f_1$  and  $f_2$  are 0-interactive with each other.

#### 4. Logistic Regression

The set of parameters  $F$  produced by the two-phase parameter selection algorithm contains most of the information. Naturally, the next step is to quantitatively measure the influences of the selected parameters. Logistic regression [SAS94, MS01] technique is employed to fit the statistical model.

Consider a target parameter  $t$  ( $t$  is a system state descriptor) that has  $l$  values ( $v_1, v_2, \dots, v_l$ ) ( $p_i$  ( $1 \leq i \leq l$ ) is the probability that  $t$  takes value  $v_i$ ) and a set of  $k$  explanatory parameters  $x_1, x_2, \dots, x_k$  (each  $x_i \in F, k \leq |F|$ ). Each parameter  $x_i$  is considered critical to the target parameter  $t$  by the logistic regression algorithm. A logistic regression model is

$$\log \left[ \frac{p_i}{1-p_i} \right] = \alpha + \beta_1 x_1 + \beta_2 x_2 + \dots + \beta_k x_k \quad (6)$$

Here  $\frac{p_i}{1-p_i}$  is referred to as *odds* and  $\log \left[ \frac{p_i}{1-p_i} \right]$  is referred to as the *logit* or *log-odds*. The impact of each parameter  $x_i$  can be figured out by carefully examining its relative coefficient  $\beta_i$ . This is demonstrated by the experimental study in next section. It is important to understand that each selected parameter represents a set of (interacting) parameters. System tuning can be performed by

- (1) Adding (removing) resources to adjust the most critical parameters identified by the logistic regression model.
- (2) Or adjusting parameters that are interactive or correlated to the most critical parameters identified by the logistic regression model.

#### 5. Experimental Study

To verify the proposed technique, we ran the Transaction Processing Performance Council's TPC-C [TPC01] benchmark suite on an Oracle 8i DBMS. The selection of Oracle is due to the widespread deployment of the DBMS. It is important to notice that there is nothing in the proposed technique that prevents it to be applied to other database systems. System parameter values were captured by running *sar*, *iostat* [IOSTAT] and querying Oracle internal data dictionaries. We then applied the proposed approach to the collected performance data set. The results are demonstrated to illustrate the usefulness of the proposed technique in Oracle database system tuning.

##### 5.1 Test Environment Setup

The Transaction Processing Performance Council's TPC-C benchmark suite was run on an Oracle 8i database system. This benchmark is an industry standard for evaluating the OLTP performance of a database system. The hardware consisted of a Sun Enterprise 450 Server with four 250Mhz UltraSparc processors, 2 GB of RAM, and 48 GB of storage striped across 12 disks. System load and capacity were systematically varied as shown in Table 1.

Dataset	User Factor ( $c_1$ )	Processors ( $c_2$ )
1	5	4
2	10	4
3	20	4
4	30	4
5	30	2

**Table 1.** Data sets and system state descriptors

We increased the workload on the system in a controlled fashion. In each test, the system was allowed a sufficient ramp-up time to reach a steady state. This ensured that results would not be skewed by operations that only occur at initialization time. Operating system and database performance data tools subsequently collected statistics, at 10-second intervals for a total of 600 seconds. These five data sets were finally combined to form an overall dataset  $U$ .

In the experiment, each data set has 60 observations, thus  $N=300$  and  $K=3238$ . As being depicted in Table 3, we have two system state descriptors  $c_1$  (number of users) and  $c_2$  (number of processors). After an initial examination of the data set, we found there are 2189 constant parameters. Since parameters that maintain constant value for all five experimental conditions cannot provide information to explain the experimental difference, these parameters are removed from the database. After reducing the constant parameters, there are 1049 parameters remaining to be investigated.

The proposed techniques were applied to the test data set to investigate the performance of the system.

First, 170 clusters were generated from 1049 non-constant parameters by variable clustering procedure. 90% of the parameters fall in the first 110 clusters. By analyzing the clusters, we were able to draw some interesting conclusions. For example, we found that Oracle achieves good scalability under various workloads by efficiently manipulating tablespaces. We also identified a bottleneck in the system optimization process through careful investigation of a particular correlation. Parameter selection was performed based on the steps introduced in Section 3.1. The threshold to select low  $r^2$ -value parameter was 0.75. Around 22% (238) of the 1049 parameters were selected from the 170 clusters generated by variable clustering.

Second, the *Regression\_Tree\_Analysis* algorithm was performed on the selected parameters. The algorithm constructed a *predictability graph* of 238 nodes. 7

parameters were selected out of the 238 parameters using the *Transitive\_Reduction* algorithm.

Finally, logistic regression technique was applied to the selected parameters. The output of the logistic regression manifested a bottleneck within a hard disk. The existence of the bottleneck significantly degraded the I/O performance of the system. Removal of the bottleneck resulted in a significant improvement in service latency.

## 6. Concluding Remarks

Traditional system tuning techniques rely heavily on human experts. Typically, hypotheses must be made about system performance before graphing tools can be used to confirm the suspected problems. This approach is tedious, time consuming, and usually incomplete because the number of parameters must be dealt with is enormous. To address these drawbacks, the main contribution of this paper is to apply statistical methods to automate system analysis. Our technique generates information to guide the user on system tuning. Our methodology is as follows:

1. A 2-phase parameter reduction technique is performed to reduce the large number of parameters to a small, but still “complete” set.
2. A statistical model is then built to quantify the effect of different system changes on the selected parameters.
3. Finally, system tuning is done by adjusting the most critical parameters revealed in the statistical model.

Future work includes automating situation (2) presented in Section 4. We also plan to enhance our analysis tool to make it useful to a wide range of applications including general knowledge discovery tasks. Investigation of other suitable statistics and data mining techniques to further improve the effectiveness of our system is underway.

## References

- [AGIS92] Rakesh Agrawal, S. Gosh, T. Imielinski, B. Iyer, Arun Swami, “An Interval Classifier For Databases Mining Applications”, Proceedings of the 1992 Very Large Databases Conference.
- [AIS93] Rakesh Agrawal, T. Imielinski, Arun Swami, “Mining Association Rules Between Sets of Items In Large Databases”, Proceedings of the ACM SIGMOD Conference on Management of Data.
- [AFS93] Rakesh Agrawal, C. Faloutsos, Arun Swami, “Efficiency Similarity Search in Sequence Databases”, Proceedings of the Conference on Foundations of Data Organization”.

- [AS94] Rakesh Agrawal, Ramakrishnan Srikant, "Fast Algorithms for Mining Association Rules", Proceedings of the 20<sup>th</sup> VLDB Conference.
- [AS96] Rakesh Agrawal, Ramakrishnan Srikant, "Mining Quantitative Association Rules In Large Relational Tables", Proceedings of the 1996 ACM, SIGMOD International Conference on Management of Data.
- [AGGR98] R. Agrawal, J. Gehrke, D. Gunopulos, P. Raghavan. "Automatic Subspace Clustering of High Dimensional Data for Data Mining Applications", Proceedings of ACM SIGMOD Conference on Management of Data, 1998.
- [B01] Don Burleson, "Oracle High Performance Tuning with Statspack", Osborne McGraw-Hill, 2001.
- [BFOS98] Leo Breiman, Jerome H. Friedman, Richard A. Olshen, Charles J. Stone, "Classification And Regression Trees", Chapman & Hall, 1998.
- [EK77] J. Eve and R. Kurki-Sunonio, "On Computing the Transitive Closure of A Relation", Acta Informatica 8, 303-314, 1977.
- [F90] K. Fukunaga, "Introduction to Statistical Pattern Recognition", Academic Press, Second Edition, 1990.
- [G01] Ken Gottry, "Successful Solaris™ Tuning", SysAdmin, the Journal for UNIX Systems and Administrators, 2001.
- [HDY99] Jiawei Han, Guozhu Dong, Yiwen Yin, "Efficient Mining of Partial Periodic Patterns in Time Series Database". Proceedings of 15<sup>th</sup> International Conference on Data Engineering, 1999.
- [HS95] Maurice Houtsma, Arun Swami, "Set-Oriented Mining for Association Rules in Relational Databases", 11<sup>th</sup> International Conference on Data Engineering.
- [INSTR01] Instrumental Inc. PerfStat/PerfAlert, <http://www.instrumental.com/software/ps2.html>.
- [IOSTAT] UNIX man pages, <http://www.ntua.gr/cgi-bin/man-cgi?iostat+ANY>.
- [IRW93] Y.E. Ioannidis, R. Ramakrishnan, and L. Winger, "Transitive Closure Algorithms Based On Graph Traversal", ACM transactions On Database Systems 18(3), 512-576, September, 1993.
- [KR90] Leonard Kaufman, Peter J. Rousseeuw, "Finding Groups in Data – An Introduction To Cluster Analysis", Wiley Series in Probability and Mathematical Statistics, 1990.
- [MS01] Charles E. McClulloch and Shayle R. Searle, "Generalized, Linear, and Mixed Models" John Wiley & Sons Inc., New York, 2001.
- [NS93] Esko Nuutila and Eljas Soisalon-Soininen, "Efficient Transitive Closure Computation", Helsinki University of Technology, Laboratory of Information Processing Science, Technical Report TKO-B113, 1993.
- [ORCLT99] Oracle, "Oracle 8i Tuning, Release 8.1.5". Part No. A67775-01, Oracle Corporation.
- [ORCLC99] Oracle, "Oracle 8i Concepts, Release 8.1.5", Oracle Corporation.
- [R73] Anderberg M. R., "Cluster Analysis for Application", New York: Academic Press, Inc.
- [R98] Raghu Ramakrishnan, "Database Management Systems". WCB/McGraw-Hill.
- [S98] Steven S. Skiena, "The Algorithm Design Manual", [Telos/Springer-Verlag](http://www.telos.com), 1997
- [SAS94] SAS/STAT User's Guide, 4<sup>th</sup> Edition, Cary, NC: Authors.
- [SAR] UNIX man pages, "System Activity Reporter", <http://www.mcscr.olemiss.edu/cgi-bin/man-cgi?sar+1>
- [SS98] Murray R. Spiegel, Larry J. Stephens, "Schaum's Outline of Theory and Problems of Statistics, third edition". McGraw-Hill.
- [T00] Heidi Thorpe, "Oracle 8i Tuning and Administration, The Essential Reference", Addison-Wesley, 2000.
- [TPC01] Transaction Processing Performance Council, "TPC Benchmark™ C Standard Specification Revision 5.0", [http://www.tpc.org/tpcc/spec/tpcc\\_current.pdf](http://www.tpc.org/tpcc/spec/tpcc_current.pdf).